**Mini-Project 1**

1. **Aligning Prokudin-Gorskii Images**

**Aim:**

To reconstruct a color photograph from three photographs taken using blue, green and red filters.

**Implementation of alignChannels.m:**

**Steps –**

1. Selected the “Blue channel” as the anchor channel.
2. Circularly shifted the “Red channel” and “Green channel” exhaustively in the range of -15 to 15 pixels, in both horizontal and vertical directions.
3. Calculated the sum of squared distances between the circularly shifted channels and the anchor channels, where, sum of squared distances is:

where, the sum is taken over the pixel values.

1. The values of the shift, for which the SSD was minimum for a particular image, was stored in the variable *predShift.*
2. The image channels were shifted by the values stored in *predShift*, in order to align them.

Figures 1, 2 and 3 give the code these implementation steps.

**Code –**

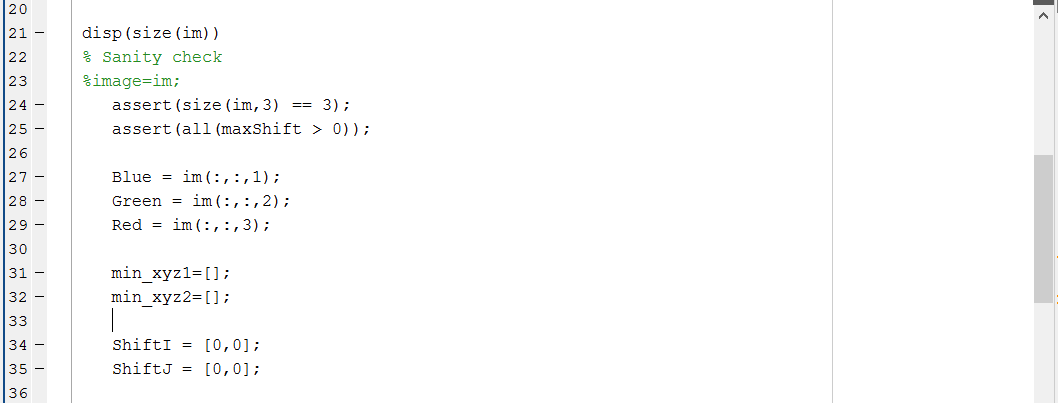


Figure : Code for alignChannels.m Part 1

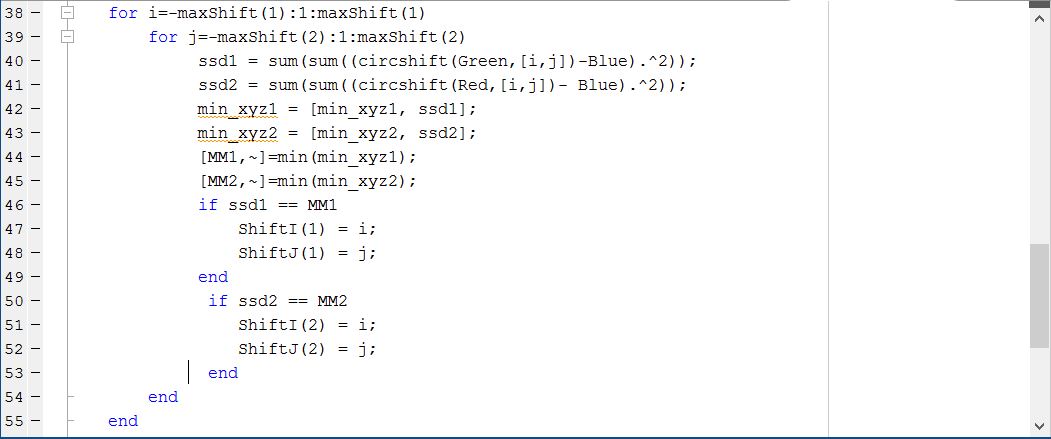


Figure : Code for alignChannels.m Part 2

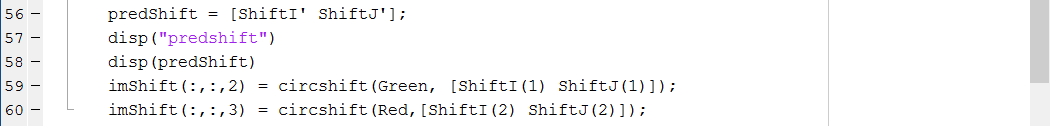


Figure : Code for alignChannels.m Part 3

**Verifying the code – evalAlignment.m**

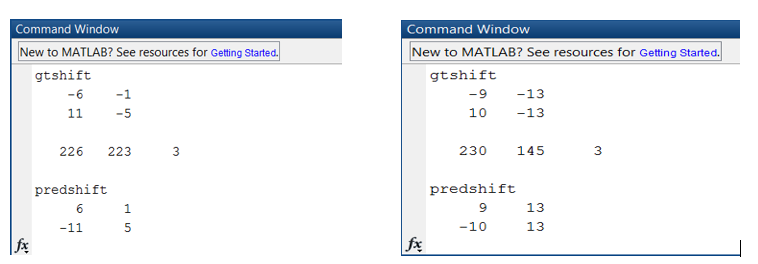
For the sake of verification, random shifts were introduced in two channels of sample images (*gtShift* in *randomlyShiftChannels.m*). The code *alignChannels.m* should be able to identify the random shifts introduced and revert them to produce align images. Intuitively, the values stored in the array *predShift* should be the negative of the values stored in the array *gtShift.* 

Figure :Examples of values in the arrays gtShift and predShift

Figure 4 show that the value of the variable *predShift* is certainly, the negative of the value of the shift generated (*gtShift*). Therefore, the alignment of the images are taking place as expected. This is depicted in Figures 5 and 6.

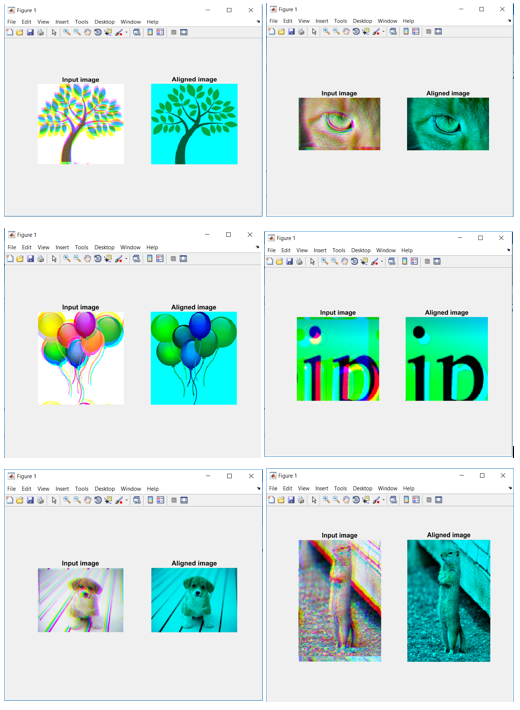


Figure : Result of evalToyAlignment.m

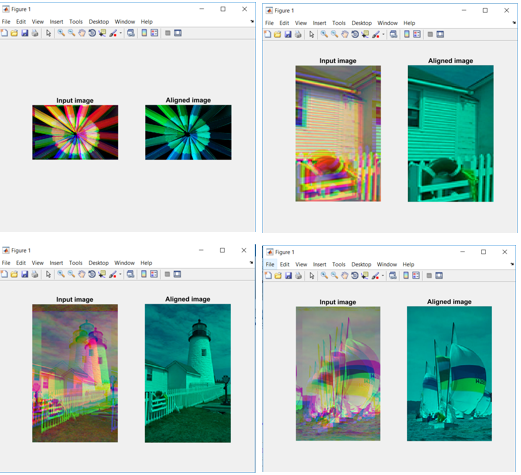


Figure : Result of evalToyAlignment.m

**Aligned Color Images from evalProkudinAlignment.m**

The following are the shift vectors and the aligned images from the Prokudin-Gorskii Image Collection.

*Table 1: Shift vectors for the red and green channels obtained using alignChannels.m for the Prokudin-Gorskii Images*

|  |  |
| --- | --- |
| **File Name** | **Shift Vectors** |
| 00125v.jpeg | (-4,1), (-10,2) |
| 00153v.jpeg | (-13,-2), (-11,-3) |
| 00398v.jpeg | (0,1), (-8,2) |
| 00149v.jpeg | (-5,0), (-9,-1) |
| 00351v.jpeg | (-9,0), (-13,1) |
| 01112v.jpeg | (-8,-1), (-8,-3) |



Figure : 00125v.jpeg

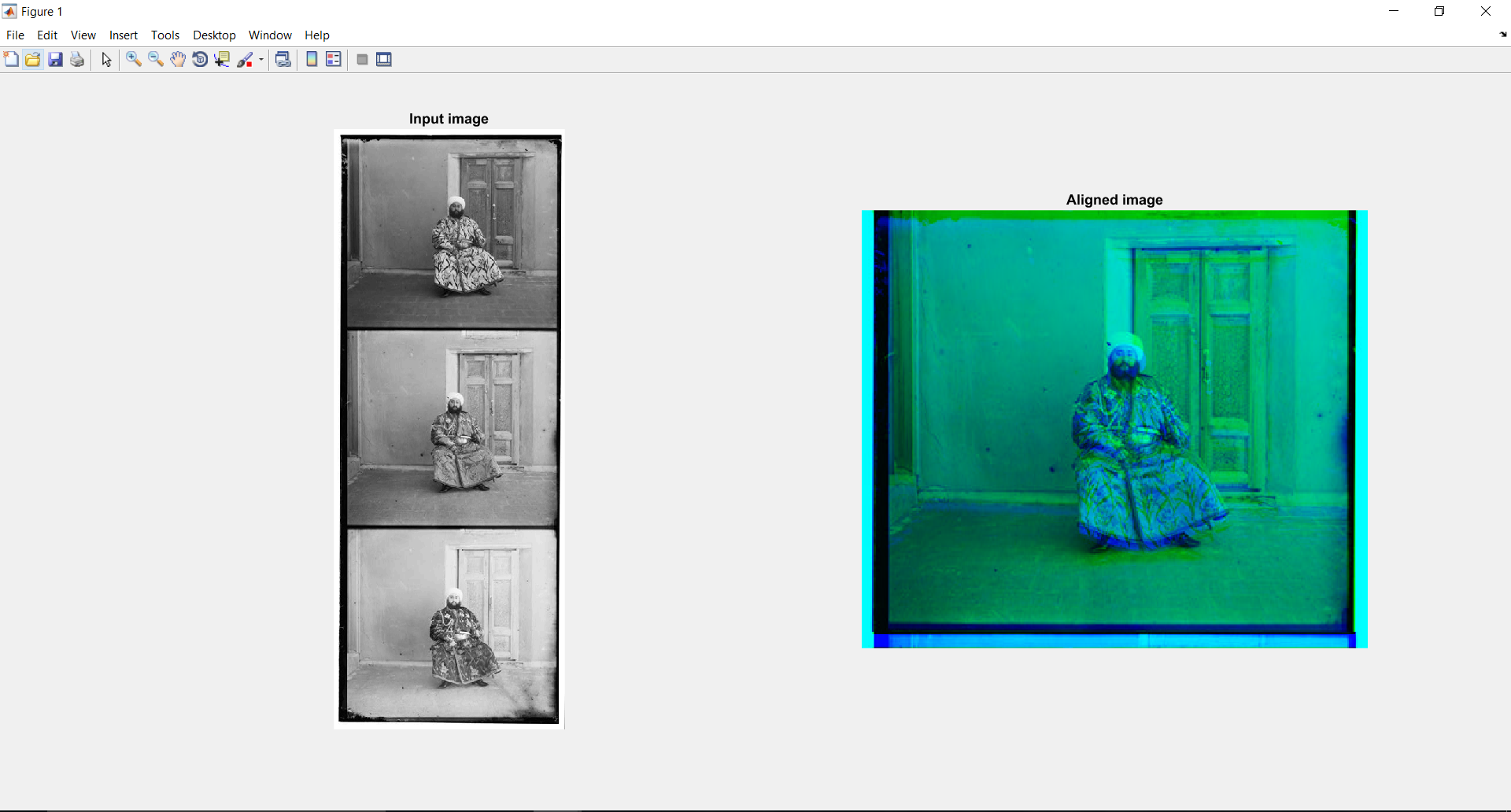


Figure :00153v.jpeg

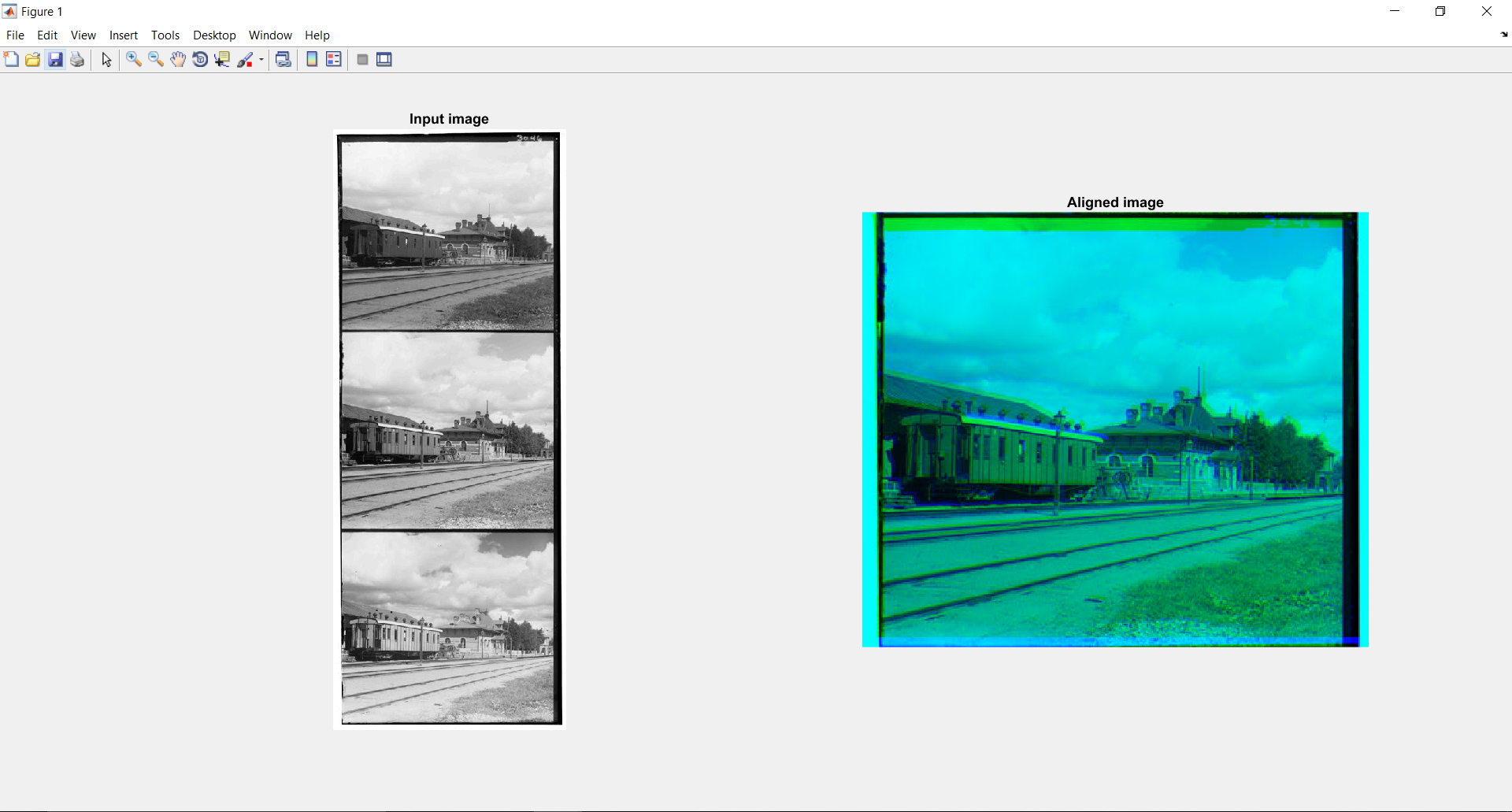


Figure :00398v.jpeg

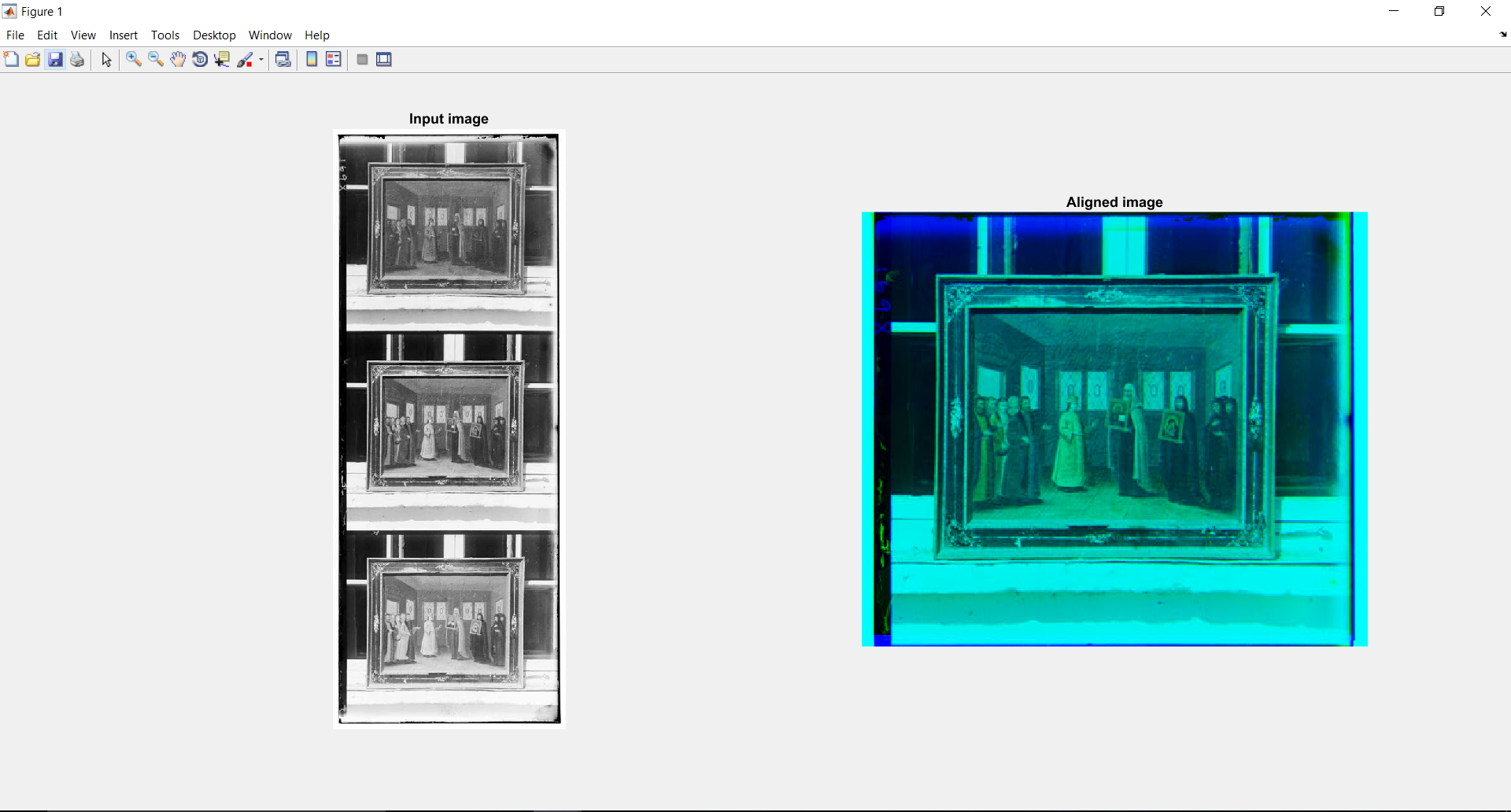


Figure : 00149v.jpeg

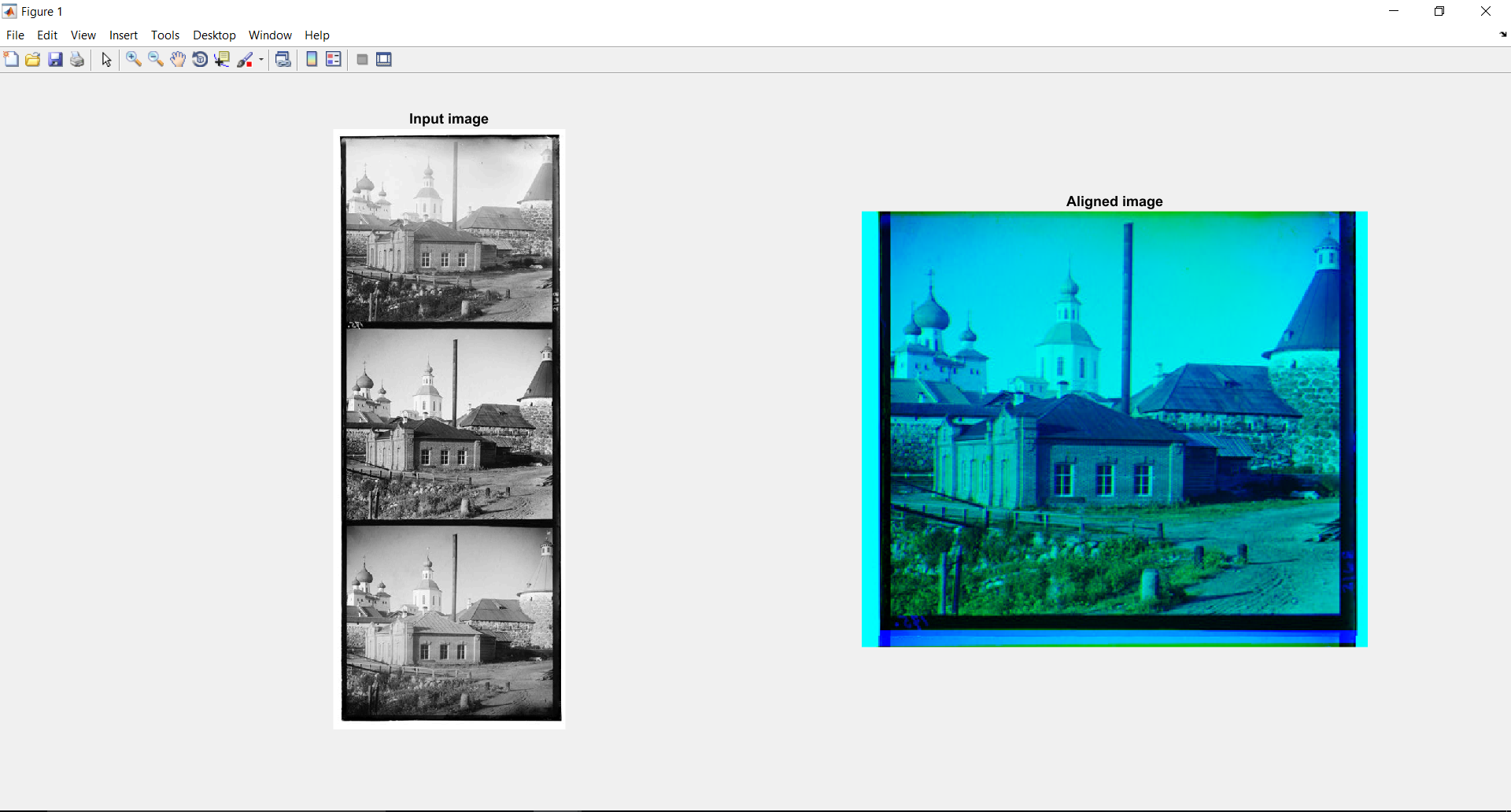


Figure :00351v.jpeg

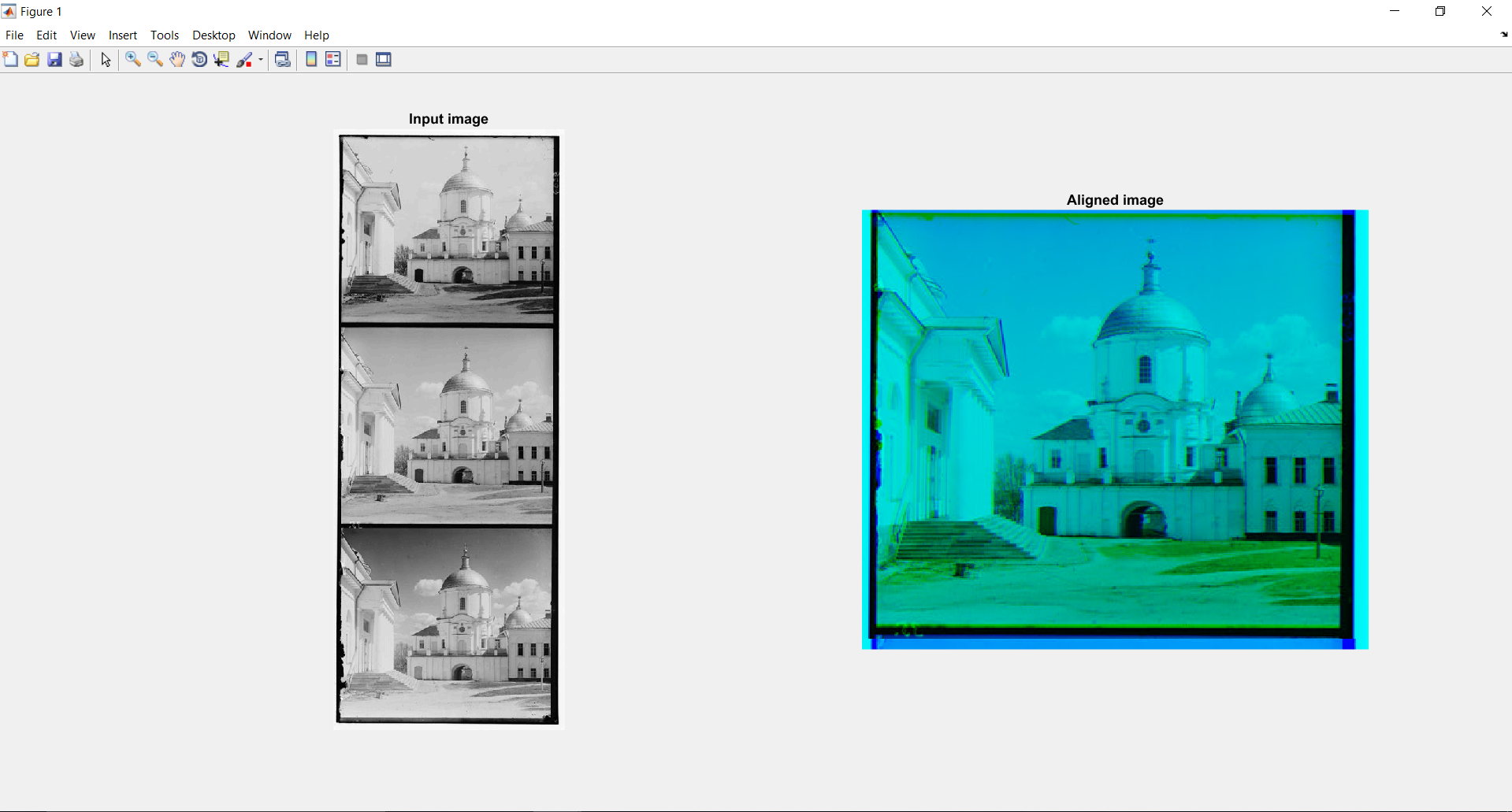


Figure : 01112v.jpeg

1. **Photometric Stereo**

**Aim:** To implement a basic shape from shading algorithm. The input to the algorithm is a set of photographs taken with known lighting directions and the output of the algorithm is the albedo, normal directions and the height map.

**Implementation of prepareData.m**

The data was preprocessed to subtract the effect of ambient light from all the images, as the images need for this algorithm, need to be under the illumination of only the point light source, as shown in Figure 13.

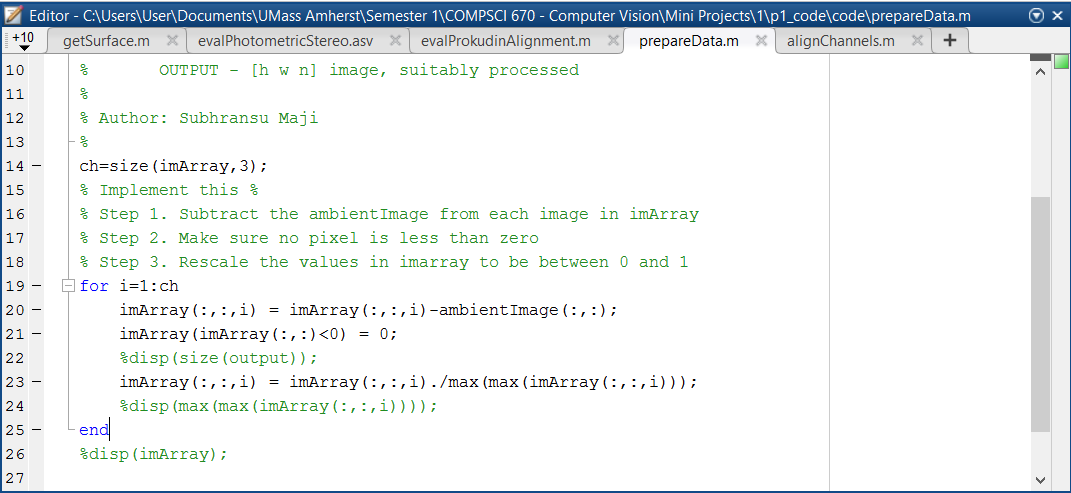


Figure : Implementation of prepareData.m

**Implementation of photometricStereo.m**

The code shown in Figure 14 and 15, calculates the Albedo and Surface Normals of a photograph, which has been captured using point light sources from different directions.

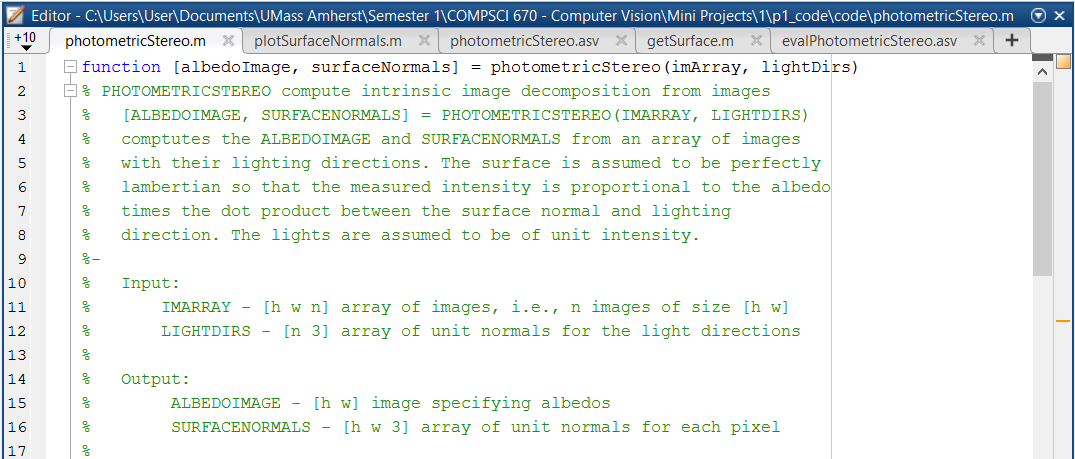


Figure : Implementation of photometricStereo.m

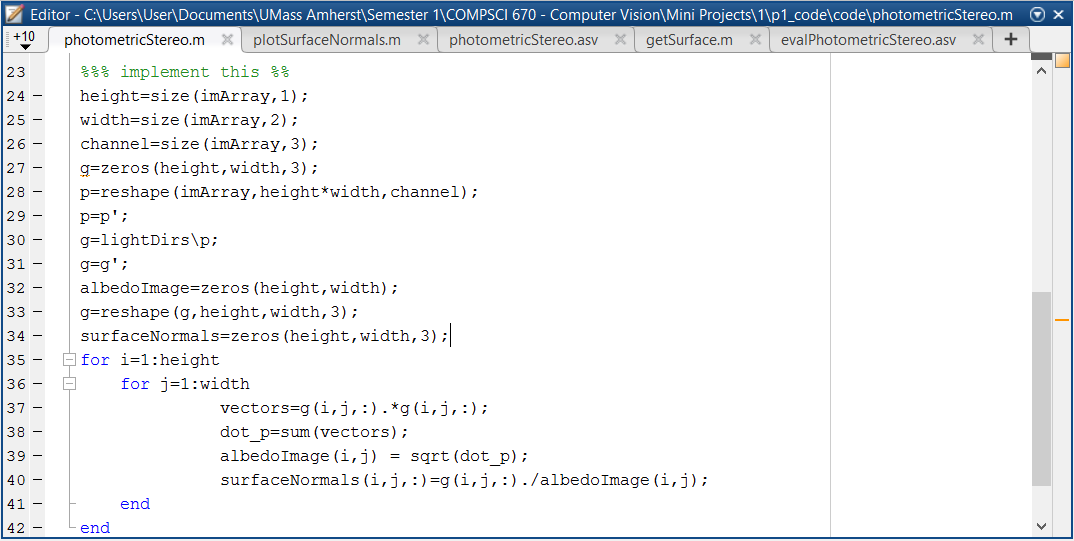


Figure : Implementation of photometricStereo.m

**Implementation of getSurface.m**

The code shown in Figures 16, 17 and 18 reconstruct the surface of the image using the surface normal retrieved from 64 different images taken in different lighting conditions.

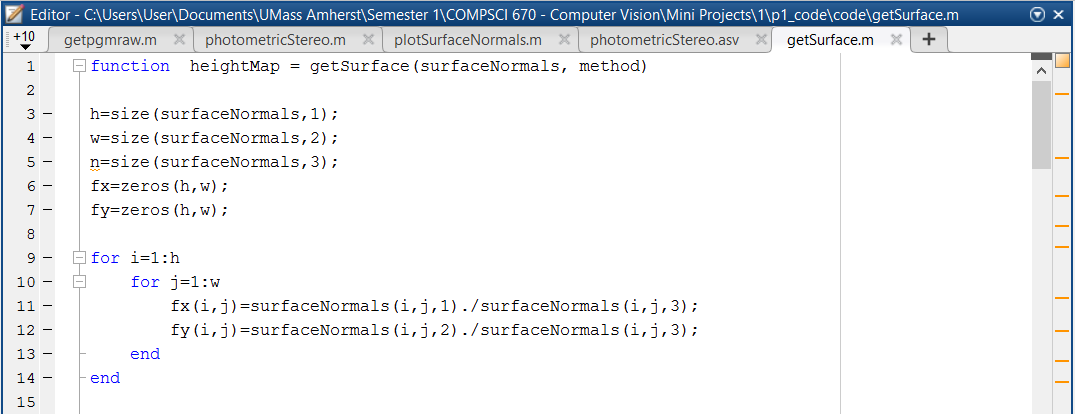


Figure : Implementation of getSurface.m

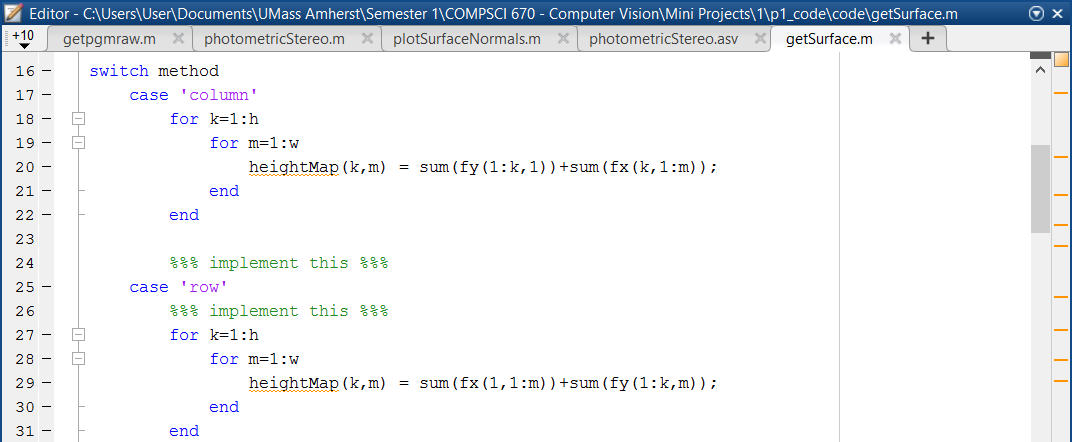


Figure : Implementation of getSurface.m

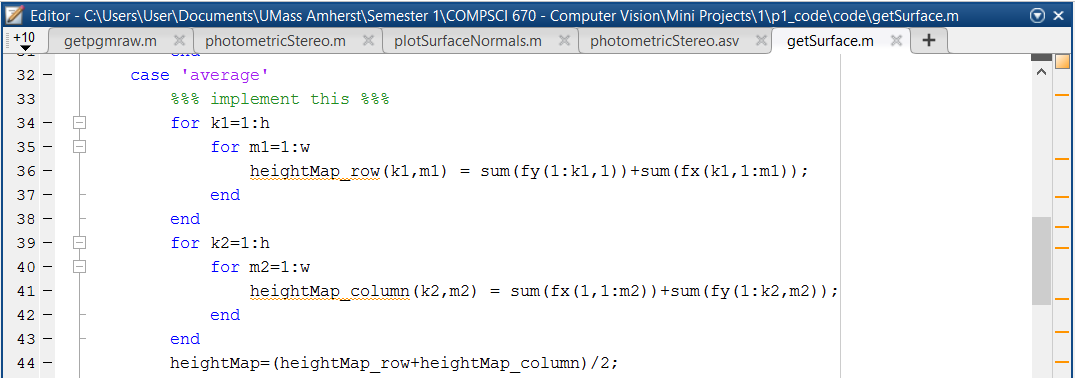


Figure : Implementation of getSurface.m

**Output**

Image: yaleB02

Integration Method: Row

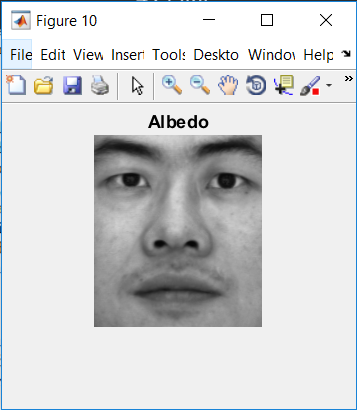


Figure : Albedo

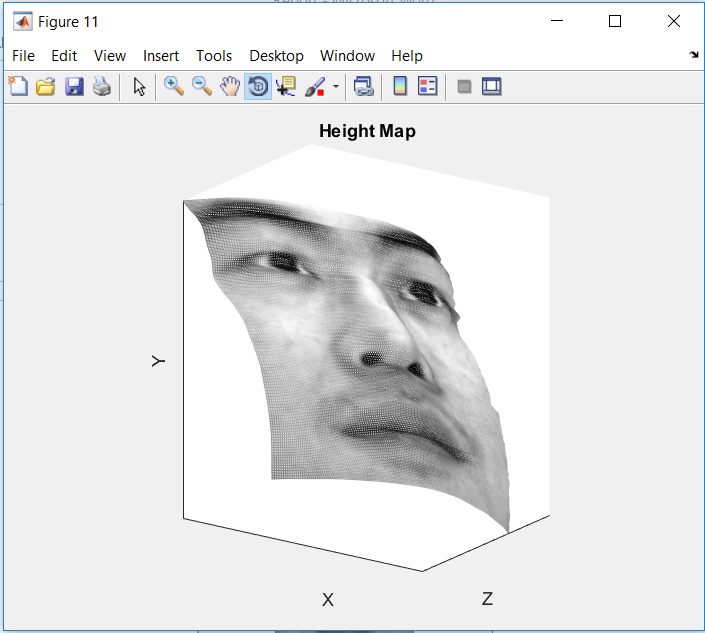


Figure : Height Map

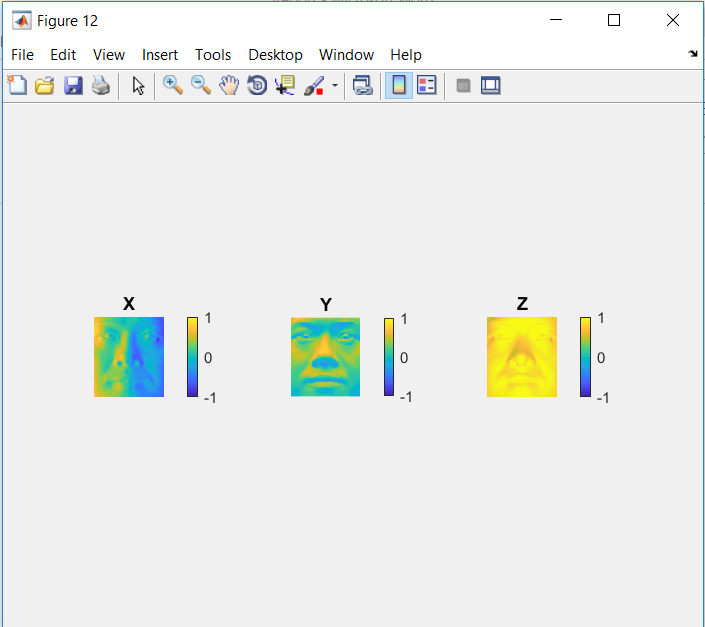


Figure : Surface Normals